CONTROLO 2008
8th Portuguese Conference on Automatic Control
University of Trás-os-Montes and Alto Douro, Vila Real, Portugal
July 21-23, 2008

"BALL AND BEAM" VIRTUAL LABORATORY: A TEACHING AID IN AUTOMATIC CONTROL COURSES

José L. Lima, José C. Gonçalves, Paulo G. Costa, A. Paulo Moreira

Department of Electrical Engineering
Faculty of Engineering of University of Porto
jllima@ipb.pt, goncalves@ipb.pt, paco@fe.up.pt, amoreira@fe.up.pt

Abstract: This paper describes an interactive learning tool that can be used in control systems lectures to a better understanding of some control methods and to improve control systems design. The Virtual Laboratory was built as a teaching aid during automatic control lectures and also to be used by students for problem solving and individual learning of different control methods. A 3D scene with a friendly appearance shows a simulation where a ball is placed on a beam being allowed to roll along its length. The Virtual Laboratory experiment goal is to stabilize the ball in a desired position changing the beam angle.

Keywords: Control education, Interactive programs, Laboratory education, Modelling, Automatic control.

1. INTRODUCTION

This paper describes a tool that supports a 3D scene simulation of the "ball and beam experiment". The ball and beam system is one of the most enduringly popular and important laboratory models for teaching control systems theory, usually used to demonstrate the dynamics of unstable systems (Wellstead, 1990). A Virtual Laboratory is composed by a PC application, reducing some hardware problems and allowing an attractive visualization (Foley, *et al.*, 2000).
The objective of this project is to provide a powerful tool that can be used in automatic control lectures to support a better understanding of some control methods and also to improve control systems design. The simulated world behaviour and graphics are based on open source platforms. The final result is an interactive 3D scene allowing the user to move around the world, as shown in Fig. 1, where a graphic shows the interest variables. Another developed Virtual Laboratory (Fig. 2), based on the same system is also presented, allowing users to make their own control programs in Pascal language. This is the main improvement since inverted pendulum virtual control laboratory was presented (Lima, 2006).

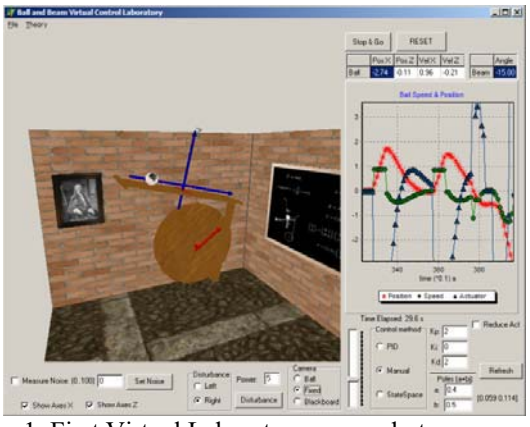

Fig. 1. First Virtual Laboratory screenshot.

This paper is organized as follows: Initially, the ball and beam overview, where the system is introduced, modelled and the world is built, is presented. Then section 3 presents the manual, PID and state-space closed loop methods where real-time interactive graphs are shown with simulation results. This section also describes both applications and its

interfaces such as user script language. Finally, section 4 rounds up with conclusions.

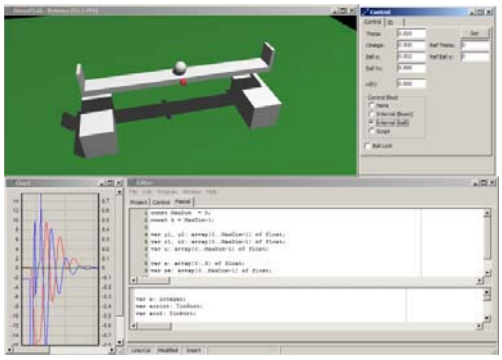

Fig. 2. Second Virtual Laboratory screenshot.

2. BALL AND BEAM OVERVIEW

The ball and beam system is widely used because it is very simple to understand as a system, and yet the control techniques that can be applied cover many important classical and modern design methods. It has a very important property: it is open loop unstable (Wellstead, 2007).

A ball is placed on a beam where it is allowed to roll with one degree of freedom along the length of the beam. A lever arm is attached to the beam at one end and a servo gear at the other. As the servo gear turns, the lever changes the angle of the beam (θ). When θ angle is different from zero, gravity causes the ball to roll along the beam. Furthermore, the beam and the controller should be able to receive a ball in any position and hold it there, or to move it according to a specification.

2.1 Open Dynamics Engine World Construction

Design a realistic behaviour without real hardware is possible due to a physics-based simulator implementation (Browning, 2003). The physics engine is the key to make simulation useful without model knowledge. Model can be calculated but just for controller achieve. The dynamic engine ODE, Open Dynamics Engine (Smith, 2000), is a powerful tool that allows programmers to create a physic world composed by objects and simulate them. It is essentially a simulation library that provides support for rigid body motion, rotational inertia and collisions treatment, where the world to be simulated is built. It also allows to use Open GL routines to render the 3D simulated environment. The Open GL routines are based on GLScene library. It provides visual components and objects allowing description and rendering of 3D scenes in an easy, no-hassle, yet powerful manner. It has grown to become a set of founding classes for a generic 3D engine with Rapid Application Development in mind (GLScene, 2000).

The ODE developed dynamic world is composed by a sphere, a beam fixed in one end (allowing the ball to roll) and a servo motor (whose angular position can be controlled).

The world construction is based on objects. Each object has properties like size, position, colour, texture and behaviour. Physical connections between different objects can be achieved by resorting to the joint capability. Joints, in this system, are slider and hinges. Slider does not mean that friction is despised, but means that an object just moves along a direction, like the ball placed on the beam. A hinge that works like a folding allows one or two objects to move through an axle.

2.2 System Modelling

Several controllers will be designed for this system so that the ball's position can be manipulated, through beam angle or beam torque, even with a collision disturbance. For controllers design, the system model is required. The modelled system scheme is shown in Fig. 3. For this problem, let's assume that the ball rolls without slipping.

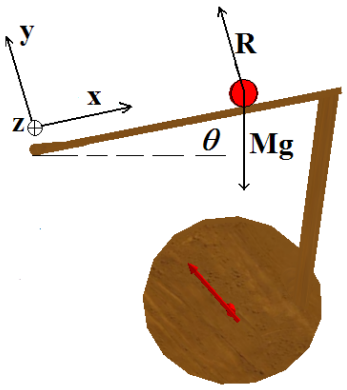

Fig. 3. Ball and Beam schematic.

The constants and variables for this case study are defined as follows; however, user can change the mass of the ball *M* and its radius *r*.

M - mass of the ball 0.125 *kg*
r - radius of the ball 0.5 *m*
g - gravitational acceleration 9.8 $m.s^{-2}$
J - ball's moment of inertia $J=2/5.M.r^2$
θ - beam angle coordinate
R - Reaction Force
$\boldsymbol{x}$ - ball position coordinate
$\dot{\boldsymbol{x}}$ - Ball speed
$\ddot{\boldsymbol{x}}$ - Ball acceleration
$\mathbf{M_{eq}}$ - Equivalent mass (ball rolls)

The well known Newton's second law of motion, equation 1, allows to write the Lagrangian equation for the described system (Michigan, 1997), where *F* is the force applied to the body and *a* is its acceleration. Equation 2 describes the motion for the ball.

$$F=Ma \tag{1}$$

$$-Mg\,sin(\theta) = -\left(\frac{J}{r^2}+M\right)\ddot{x} \tag{2}$$

Small angles (Hauser, 1992), where $sin(\theta)\approx\theta$, allow to linearize the equation 2 about $\theta\approx 0^o$ resulting in

equation 5 where equation 4 represents the equivalent mass because the ball rolls and the moment of inertia cannot be despised (Márquez, 2003). Let u be the system input (motor angle control) in equation 3.

$$u \approx (\theta) \tag{3}$$

$$M_{eq} = \frac{J}{r^2} + M \tag{4}$$

$$\ddot{x} = \frac{Mg}{M_{eq}} u \tag{5}$$

The linearized system equations can also be represented in state-space form (Ribeiro, 2002), done by selecting the ball's position x and velocity $\dot{x}$ as the state variables and the gear angle u as the input. The state-space representation is shown in equation 6 and equation 7 where Y is the system output (ball position).

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{Mg}{M_{eq}} \end{bmatrix} \tag{6}$$

$$Y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{7}$$

It is possible to use the constants (M, g and M_{eq} previously presented), that results in the state-space numerical representation, as shown equation 8.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 7 \end{bmatrix} u \tag{8}$$

Linearization is used for controller achieves while simulation is based on real model with all its nonlinearities.

3. REAL TIME INTERACTIVE SIMULATION

The control of unstable systems is critically important to many of the most difficult control problems and must be studied in the laboratory (Wellstead, 2007).

The control job is to automatically regulate the ball position by changing the angle of the beam. This is a difficult control task because the ball does not stay in one place on the beam, but moves with an acceleration that is proportional to the tilt of the beam. In control theory this system is open loop unstable because its output (ball position) increases without limit for a fixed input (angle deviation) (Iqbal, *et al.*, 2005). Feedback control must be used to keep the ball in a desired position.

Assuming that simulation time step is much faster than dynamics, it is possible to assure continuous modeling to simulate this system. A time step of 20 ms is enough to validate this approach (much faster than system dynamics). Both presented laboratories have the same 50 Hz frequency.

There are three embedded closed-loop control methods and a script program possibility, presented in next subsections, which are used in this project: The manual control, where user can control the beam angle with a slider, the PID control where user can introduce proportional, integral and derivative gains and the state-space feedback control where user can apply pole placement. The embedded script gives user the freedom to create any controller.

A graphical time evolution shows the ball position, speed and the desired control angle where a zoom feature is also implemented in order to highlight some details. The control system simulation starts running at zero seconds for the ball placed still at x=0 and zero degrees for beam angle. System remains in its initial conditions until a disturbance (ball collision) or a slope occurrence.

3.1 Manual control

This control "method" allows users to change a slide bar position with mouse that slopes the beam controlling the ball position.

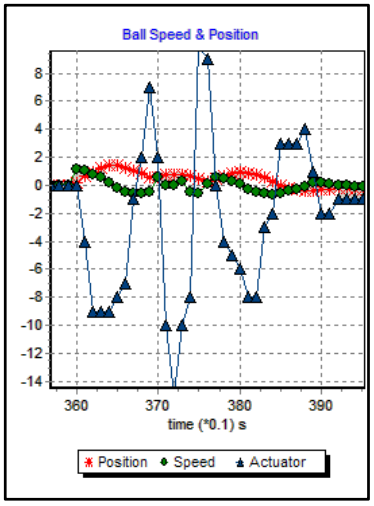

Fig. 4. Manual Control method data graph.

It is a difficult task and result depends on the user's sensibility. Figure 4 presents an example where a disturbance in ball position happens at t=36s and the user reaction is shown. A collision with another ball is used to disturb the system.

3.2 PID controller

The PID control method has been successfully used in many industrial control systems for over half a century. The basic principle of the PID control scheme is to act on the variable to be manipulated through a proper combination of three control gains: proportional (K_{Pr}), integral (K_i) and derivative (K_d) as presented in equation 9. In its most simple form, PID involves three mathematical control functions working together as a function of error signal, *e(t)*, (desired and actual output deviation).

$$u(t) = K_{pr}e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \quad (9)$$

As a proportional controller does not make the system stable, a PD closed loop controller was chosen as an example and can be designed as presented in Fig. 5, where K_P is the plant gain described in equation 10, K_C is the controller proportional gain and T_d the derivative gain.

$$K_p = \frac{M_{eq}}{Mg} \quad (10)$$

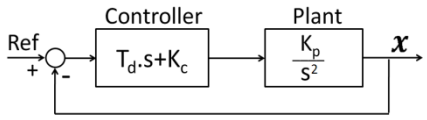

Fig. 5. PD closed loop system.

The closed loop transfer function is shown in equation 11, where *Ref(s)* is the desired ball position.

$$\frac{X(s)}{Ref(s)} = \frac{K_p(T_d \cdot s + K_c)}{s^2 + K_p \cdot T_d \cdot s + K_c \cdot K_p} \quad (11)$$

Gains T_d and K_c can be found by equations 12 and 13, as a second order system response, where ζ is the damping coefficient and ω_n is the natural frequency for the desired conditions (Carvalho, 1993): time settling (T_s) and percentage overshoot (PO) presented in equations 14 and 15.

$$T_d = \frac{2 \cdot \zeta \cdot \omega_n}{K_p} \quad (12)$$

$$K_c = \frac{\omega_n^2}{K_p} \quad (13)$$

$$PO = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \quad (14)$$

$$T_s = \frac{4}{\zeta\, \omega_n} \quad (15)$$

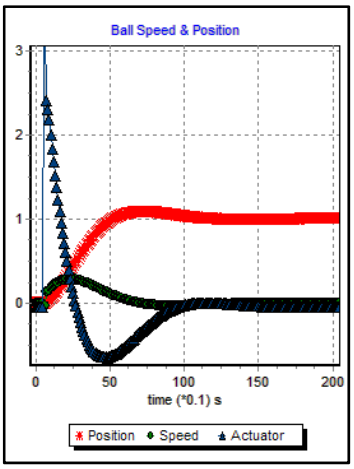

Fig. 6. PID control method data graph.

The presented equations can be used to find K_p and T_i for an example of a 10% of PO and a T_s of 10 seconds that gives Kc=3.204 and K_d=5.598 gains.
User can manually chose PID gains, comparing different performances. The data acquisition of the closed loop system example of a unitary step is shown in Fig. 6 and ball position result accordance with desired conditions allows to validate the proposed simulator.
Another procedure for choosing closed-loop pole locations for an *nth*-order plant is based on Bessel prototype. The first step is to choose the desired settling time of the closed loop system based on performance specifications, taking in account the limitations of the system hardware. Then, the roots of the nth-order normalized Bessel polynomial must be divided by *Ts* to obtain the desired closed loop locations. The poles *-4.0530 ± j2.3400* are the roots of a second order normalized polynomial corresponding to a settling time of one second (Vaccaro, 1995).

3.3 State-space controller

Having the system model described in state-space, the *K* values (closed loop gains) can be found by: *A-BK* eigenvalues (Ogata, 2002).
The desired closed loop poles are presented in equation 16 and as result the state-space feedback vector is given by equations 17 and 18.

$$poles = a \pm ib \quad (16)$$

$$k_1 = \frac{2}{7}a \quad (17)$$

$$k_2 = \frac{a^2+b^2}{7} \quad (18)$$

The closed loop diagram system is shown in Fig. 7 where *Ref.* is the desired ball position, *u* is the system input, *Y* the system output (ball position) and *X* the state.

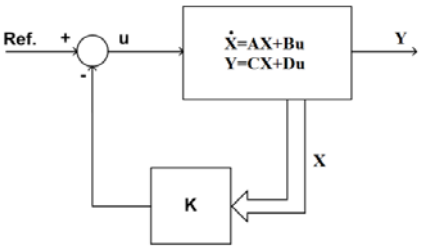

Fig. 7. State-space schematic.

As an example, let's suppose the same conditions of the PD previously presented example, a T_s of 10 seconds and a PO of 10%. The system transfer function, presented in equation 19, allows to find the T_s and PO, presented in equations 14 and 15, like in previous example.

$$\frac{X(s)}{Ref(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (19)$$

For the desired *Ts* and *PO*, the state-space controller data is shown in Fig. 8 where, in this example, *a=0.4* and *b=0.5416*, being the state vector represented by equation 20.

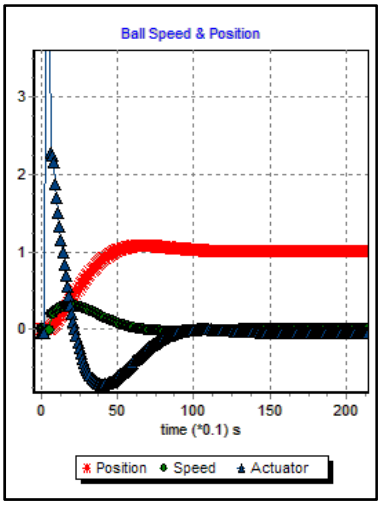

Fig. 8. Fast state-space controller data graph.

$$K = [0.065 \quad 0.114] \tag{20}$$

This is a fast controller forcing the ball to lose contact with the beam, as shown in Fig. 9, due to the fast lever arm movements, as shown in Fig. 10, where a detailed scale data graph shows this nonlinear issue from 0.3 to 0.8 seconds.

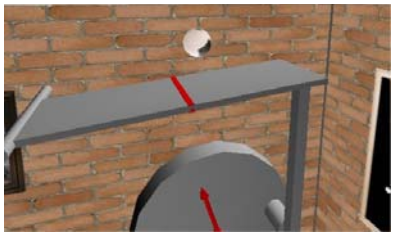

Fig. 9. Ball detached from the beam.

While ball is detached from the beam, the controller actions are useless, reducing its performance. When ball touches the beam, controller recovers its handling.

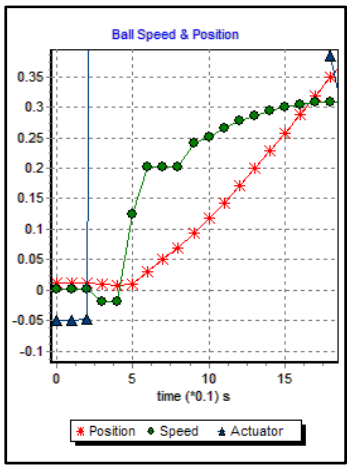

Fig. 10. Detailed data graph during ball detached from the beam phenomena.

As another example, a slower controller, for a T_S of 15 seconds and an *PO* of 20%, presented in Fig. 11, is implemented where, in this example, *a*=0.2667 and *b*=0.3638, being the state vector represented by equation 21.

$$K = [0.029 \quad 0.076] \tag{21}$$

The ball never loses touch with the beam, but for the same input reference step, the ball is stabilized slower. As it can be observed in Fig. 8 and 11, the slower system takes at least 5 more seconds to stabilize the ball in the desired position.

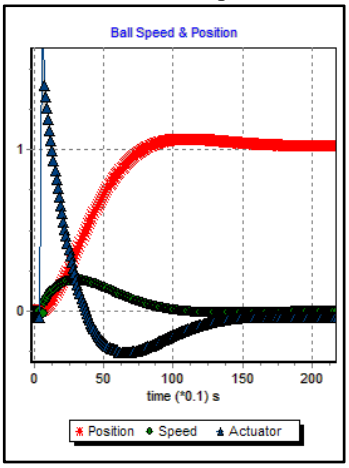

Fig. 11. Slow state-space controller data graph.

3.4 Noise disturbance feature

Some problems arise in physical systems data acquisition such as signal noise, cumulative errors, electromagnetic interference and temperature drift. Noise addition in ball positioning measurement allows to illustrate this problem. Gaussian noise can be added in ball position and speed measure with desired amplitude.

The data acquisition graph, where a noise disturbance is introduced between *16.5* and *19.5* seconds, is presented in Fig. 12 showing the controller response (poles *a*=0.4 and *b*=0.5416) for the desired conditions presented in the previous faster example.

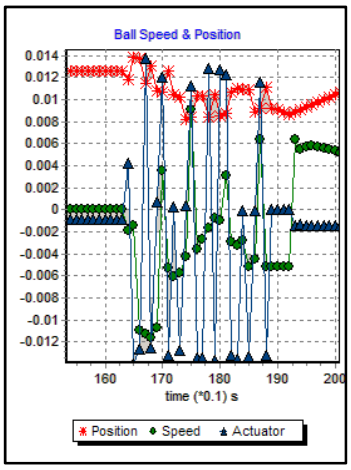

Fig. 12. Noise addition data graph.

3.5 Script language control method

In order to allow users to create their own control programs, a script window that accepts Pascal language code was developed. Users can try several no predefined controllers where real-time results are presented.

The developed control procedure can be applied to a real system, based on a microcontroller. There are some Pascal compilers to embedded systems such as Embedded Pascal (2004) and E-Lab (2003).
The simulator provides to the user the following information:

y1[k] - Beam actual angle (*rad*)
y2[k] - Ball position x axle (*m*)
r1[k] - Beam reference angle (introduced by user)
r2[k] - Ball reference position (introduced by user)
u[k] - Torque control signal (*N.m*)

There are also the last seventh recorded samples as presented:

- *y1[k], y1[k-1], ..., y1[k-7]*
- *y2[k], y2[k-1], ..., y2[k-7]*
- *r1[k], r1[k-1], ..., r1[k-7]*
- *r2[k], r2[k-1], ..., r2[k-7]*
- *u[k], u[k-1], ..., u[k-7]*

The fourth order system state composed by *y1[k]*, *y2[k]* and their derivatives are recorded too.
As an example, a proportional controller Pascal procedure is presented on the lower side of Fig. 13 for the beam angle regulation.

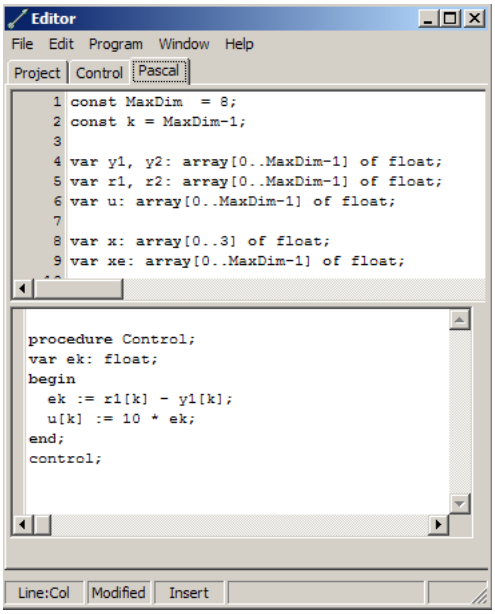

Fig. 13. Pascal Script program example.

The *e[k]* variable is actualized each sampling time (50 Hz rate) as the error between reference and beam angle. The applied torque to the beam is K_p times *e[k]*.
As a complex example, user can develop an auto tune controller (Yu, 2006).

4. CONCLUSIONS

This paper has introduced an interactive learning tool for automatic control courses, where it is desired to control a ball position on a beam in a Virtual Laboratory.
The 3D scene visualization and animation effects are an advantage for students, giving them a better understanding of the physical systems.
The real time graphics, where ball speed, ball position and controller angle are shown, present data in a way that can be easily decoded by students. The Pascal script code can be programmed by user allowing the freedom of control method choice.

REFERENCES

Browing, B. and E. Tryzelaar (2003). Übersim: A Multi-Robot Simulator for Robot Soccer *Conference on Autonomous Agents and Multi-Agent Systems*, Australia.
Carvalho, J. L. Martins (1993). *Dynamical Systems and Automatic Control*, Prentice Hall.
Embedded Pascal (2004). http://users.iafrica.com/r/ra/rainier/pbody.htm.
E-LAB Computers (2003). http://www.elab-pascal.de/index_en.html.
Foley, J. D., A. Van Dam, S. K. Feiner and J. F. Hughes (2000). *Computer Graphics: Principles and Practice in C*, Addison-Wesley, United States of America.
GLScene (2000). http://glscene.sourceforge.net.
Hauser J., S. Sastry and P. Kokotovic (1992). Nonlinear control via approximate input-output linearization: the ball and beam example, *IEEE Transactions on Automatic Control*, pp. 392-398.
Iqbal, J., M. A. Khan, S. Tarar and Z. Sabahat (2005). Implementing Ball Balancing Beam Using Digital Image Processing And Fuzzy Logic, *18th Annual Canadian Conference on Electrical and Computer Engineering*, IEEE publishes, Saskatoon, Canada.
Lima, J., J. Gonçalves, P. Costa and A. Moreira (2006). Inverted pendulum Virtual Control Laboratory, *7th Portuguese Conference on Automatic Control*, Lisbon.
Márquez, H. J. (2003). *Nonlinear Control Systems: Analysis and Design*, John Wiley and Sons, Inc., New Jersey.
Michigan Regents of the University of Control Tutorials for Matlab (1997). http://www.engin.umich.edu/group/ctm/index.html.
Ogata, K. (2002). *Modern Control Engineering,* Prentice Hall.
Ribeiro, Maria Isabel (2002). *Análise de Sistemas Lineares*, IST Press.
Smith R., Open Dynamics Engine (2000). http://www.ode.org/.
Vaccaro, J. (1995). *Digital Control: a State-Space approach*. McGraw-Hill International Editions.
Wellstead, P. E. (2007). Control Systems principles , http://www.control-systems-principles.co.uk/whitepapers/ball-and-beam1.pdf.
Wellstead, P. E. (1990). Teaching control with laboratory scale models, *IEEE Transactions on Education*, pp. 285-290.
Yu Cheng-Ching (2006). *Autotuning of PID Controllers: A Relay Feedback Approach*, Springer.